[19]中华人民共和国国家知识产权局

[51]Int. Cl[7]
G06F 1/00

[12]发明专利申请公开说明书

[21]申请号 99811086.8

[43]公开日 2001年10月17日

[11]公开号 CN 1318163A

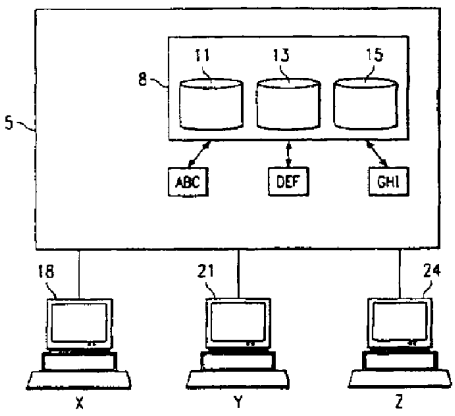
[22]申请日 1999.7.15 [21]申请号 99811086.8
[30]优先权
[32]1998.7.17 [33]US [31]09/118,621
[86]国际申请 PCT/US99/16029 1999.7.15
[87]国际公布 WO00/04435 英 2000.1.27
[85]进入国家阶段日期 2001.3.19
[71]申请人 电子资讯系统有限公司
地址 美国德克萨斯州
[72]发明人 J·N·格什菲尔德 S·G·巴杰
[74]专利代理机构 中国专利代理(香港)有限公司
代理人 吴立明 傅 康

权利要求书4页 说明书21页 附图页数2页

[54]发明名称 可选择性定义对应用程序功能部件访问的系统和方法

[57]摘要

一种方法和系统用于定义用户对一个应用程序所具有的一个或多个功能部件的访问。一个或多个"属性"分配给计算机系统(5)的用户,并存储到数据表(11,13,15)中。每一个属性拥有一个名称标明被定义访问的功能部件(例如,访问数据库中数据的权力),还有一个值定义访问的权限。属性可以成组进行分配,从而减小为每个用户单独进行属性分配而准备的负担。当一个应用程序运行,属性被检索并执行,这样就依照所检索的属性,用户对应用程序功能部件的访问被定义。



ISSN1008-4274

01·03·19

权　利　要　求　书

1. 在可以运行至少一个应用程序和保存有数据库的计算机系统中，每个应用程序至少有一个功能部件，一种定义用户对至少一个功能部件访问的方法包括以下步骤：

给所述用户分配至少一个属性；

将至少一个属性存储在数据库的第一个表中；

运行计算机系统中应用程序；

从第一个表中检索一个或多个所述至少一个分配给用户的属性；

施加检索到的属性，由此根据检索的属性定义用户对应用程序至少一个功能部件的访问。

2. 权利要求 1 中所述的方法，其中一个或多个所述至少一个功能部件与访问数据的能力相关。

3. 权利要求 1 中所述的方法，在运行步骤之前，包括以下步骤：

给每个所述至少一个属性的实际值分配父值，由此一个或多个父子关系被建立；和

将一个或多个父子值关系存储在数据库的第二个表中；

在所述施加步骤之前，该方法还包括以下附加步骤：

从第二个表中检索一个或多个父子值关系；

依据检索的父子值关系决定其中一些检索的属性是否可以被删除。

4. 在可以运行至少一个应用程序和保存有数据库的计算机系统中，每个应用程序至少有一个功能部件，一种定义用户对至少一个功能部件访问的方法包括以下步骤：

给所述用户分配至少一个属性；

将所述至少一个属性存储在所述数据库的第一个表中；

在计算机系统中运行应用程序；

从数据库中第一个表中检索一个或多个所述至少一个属性分配给用户；

提供检索的属性用于运行应用程序；和

施加属性，由此根据检索的属性定义用户对应用程序所述至少一个功能部件的访问。

5. 权利要求 4 中所述的方法，在运行步骤之前，还包括一附加步骤，根据分配给所述用户的至少一个属性，向用户提供运行一个或多个应用程序的选择。

6. 权利要求 4 中所述的方法，其中一个或多个至少一个功能部件与访问数据的能力相关。

7. 权利要求 4 中所述的方法，在运行步骤之前，包括以下附加步骤：

给每个所述至少一个属性的实际值分配父值；

将父子值关系存储在数据库的第二个表中；

在施加步骤之前，该方法还包括以下附加步骤：

从第二个表中检索父子值关系；

依据检索的父子值关系决定检索的属性是否可以被删除。

8. 在可以运行至少一个应用程序和保存有数据库的计算机系统中，每个应用程序至少有一个功能部件，一种定义用户对所述至少一个功能部件访问的方法包括以下步骤：

给组分配至少一个属性；

将所述组分配给至少一个用户；

将组存储在数据库的一个表中；

在计算机系统中运行应用程序；

从数据表中检索分配给用户的组；和

施加属性，由此根据从表中检索分配给组的至少一个属性定义用户对应用程序至少一个功能部件的访问。

9. 权利要求 8 中所述的方法，其中给所述组分配的至少一个的属性定义仅对所述应用程序的访问。

10. 一种计算机系统包括：

运行应用程序的装置，该程序至少有一个功能部件；

维护数据库的装置；

给用户分配至少一个属性的装置；

在数据库的第一个表中存储至少一个属性的装置；

从所述第一个表中检索给用户分配的所述至少一个属性的装置；和

施加属性的装置，由此根据分配给用户的至少一个的属性定义用

户对应用程序至少一个功能部件的访问。

11. 权利要求 10 中所述的系统，其中一个或多个所述至少一个功能部件与访问数据的能力相关。

12. 权利要求 10 中所述的系统进一步包括：

给每个所述至少一个属性的实际值分配父值的装置，由此一个或多个父子关系被建立；

在数据库第二个表中存储一个或多个父子值关系的装置；

在第二个表中检索一个或多个父子值关系的装置；

依据检索的父子值关系决定其中一些检索的属性是否可以被删除的装置。

13. 一种计算机系统包括：

运行应用程序的装置，该程序至少有一个功能部件；

保存数据库的装置；

给用户分配至少一个属性的装置；

在数据库的第一个表中存储至少一个属性的装置；

从第一个表中检索给用户分配了至少一个属性的装置；

向应用程序提供检索的属性的装置；和

施加属性的装置，由此根据分配给用户的至少一个属性定义用户对应用程序至少一个功能部件的访问。

14. 权利要求 13 中所述的系统，进一步包括根据分配给用户的至少一个属性，向用户提供运行一个或多个应用程序的选择的装置。

15. 权利要求 13 中所述的系统，其中一个或多个至少一个功能部件与访问数据的能力相关。

16. 权利要求 13 中所述的系统进一步包括：

给每个至少一个属性的实际值分配父值的装置；

在数据库第二个表中存储父子值关系的装置；

在第二个表中检索父子值关系的装置；

依据检索的父子值关系决定其中一些检索的属性是否可以被删除的装置。

17. 计算机系统包括：

运行应用程序的装置，该程序至少有一个功能部件；

保存数据库的装置；

给组至少分配一个属性的装置；

给用户分配组的装置；

在数据库的第一个表中存储组的装置；

从数据表中检索给用户分配的组的装置；和

施加检索的属性的方法，由此根据从数据表中检索的至少分配给组的一个属性定义用户对应用程序至少一个功能部件的访问。

18. 权利要求 17 中所述的系统，其中为组分配的所述至少一个属性定义仅对应用程序的访问。

说　明　书

可选择性定义对应用程序功能部件访问的系统和方法

发明的技术领域

本发明一般涉及定义用户对计算机系统的访问，特别是多数用户对计算机系统中一个或多个可运行程序的功能部件进行访问时，具有选择性适应性地定义每一个用户的访问限制。

发明背景

在例如具有资源共享服务的部门环境的环境中，其中多个雇员和/或客户可以访问允许运行多个应用程序的计算机系统，这就需要可以依据特定用户或用户类别来限制对于应用程序一个或多个功能部件的访问。这里用到的术语“功能部件”包括了几乎所有可能的程序功能，举例说明，存取数据库中的数据，产生、查看和打印报告，以及发送和/或接受电子邮件。

目前，对于限制用户访问的适应性还未实现。在限制访问数据方面，最近 Oracle 公司在其数据库程序中使用了一种方法，用来在数据库水平上限制访问特殊数据表的用户权力。Oracle 公司实现这一方法是通过给用户分配“角色”来限制对含有数据的表格访问，而不是对数据本身的访问。

限制对应用程序功能部件的访问，包括由 Oracle 角色限制的数据访问功能部件，需要更好的适应性，通过一个简单的例子来说明。以下是一张假设的数据表格，显示的是客户 A、B 和 C 在 1998 年 6 月 15 日上午进行的机密的金融交易，其中 WDRWL 表示提款，DPST 表示存入， PYMNT 表示支付。

表 1

	客户	时间	类型	金额
1	A	9:15A	WDRWL	1000.00
2	B	9:17A	DPST	2500.00
3	B	9:24A	DPST	1750.00
4	A	9:35A	PYMNT	5000.00
5	C	10:02A	WDRWL	50.46
6	A	10:41A	DPST	106.08
7	C	10:47A	PYMNT	530.06

为了准备一份只是关于6月份客户A的机密交易报告，需要读取1、4、6行的数据，而不需要2、3、5、7行的。由于这一数据具有很大的敏感性，对于数据的读取仅限制在与任务（即客户A的交易报告）有关的范围内是非常必要的。

此外，用于准备A过去交易报告的应用程序可能具备产生不同类型报告的能力，包括除了显示过去的表现，还有计划未来任务的报告。依据被分配任务的对象，可能并不希望准许访问两种类型的生成报告。可能也不希望准许打印生成报告。

发明概要

根据本发明，一个或多个“属性”分配给可运行多个应用程序的计算机系统的用户。每一个属性是一名称-值对，其中名称标明应用程序功能部件或是被定义访问的功能部件（例如，读取数据，产生报告），值设置访问权限（例如，全部或部分数据）。属性可以成组分配以减少准备为每一个用户进行单独的属性分配的负担。

根据本发明，提供一种系统和方法用以定义至少运行一个应用程序功能部件的用户权力。依照该系统和方法，用户至少分配一个属性。属性存储在数据库的表格中。应用程序被用户运行，然后分配给用户的属性被检索。属性通过程序被执行，这样依据所检索的属性，定义用户对于应用程序功能部件的访问。

根据本发明进一步的特征，提供用以定义至少运行一个应用程序功能部件的用户权力的系统和方法，其中组至少分配一个属性，而组分配给用户。组存储在数据库的表格中。应用程序被用户运行，而后分配给用户的组被检索。分配给组的属性通过程序被执行，这样依据所检索的属性，定义用户对于应用程序功能部件的访问。

因此，本发明的目的是提供可选择性定义对应用程序功能部件的访问的能力，该功能部件对于计算机系统中给定的用户或用户组是可利用的。

本发明的进一步目的是，在限制用户对存储在面向表格的数据库中数据访问的能力方面，提供比现有水平更强的适应性。

为更好的理解本发明以及进一步目的，参考以下结合附图的描述，它的范围将在附带的权利要求中指出。

附图的简略描述

图 1 的框图示出了依照本发明的示例性系统；

图 2 的框图示出了用户属性系统分组方案的范例；

图 3 的流程图示出了本发明方法的一个具体实例。

发明的详尽描述

用户属性

图 1 的框图示出了依照本发明的示例性系统。计算机 5 运行数据库软件 8，其保存有数据表，图示为表 11、13 和 15。许多应用程序，图示为程序 ABC、DEF 和 GHI，也在计算机 5 上运行。其中一些是用于检索和使用表 11、13 和 15 中的数据。每一个该系统的用户，X、Y 和 Z，可以通过终端访问计算机 5，图示分别为计算机 18、21 和 24。依照本发明，用户 X、Y 和 Z 分配到一个或多个“属性”。每一个属性有一个名称，标明访问被定义的应用程序功能部件（例如，访问数据库中的数据的权力），并且有一个定义访问权限的值，在后面将更详细的解释。如果不特别指出，如这里所用到的，术语属性将用来指名称-值对。

属性通过数据库软件 8 被保存在表中，并分别定义用户运行应用程序 ABC、DEF 和 GHI 的权力。例如，两个有用的属性为 DATA_SCOPE 和 USER_LEVEL。DATA_SCOPE 定义用户被允许访问的数据，并用前面表 1 作为范例，其拥有的可能的值为 A、B、C 或 ALL，它们分别表示与客户 A、B、C 或所有三个客户有关的数据。USER_LEVEL 是一较宽的属性，其通常定义用户选择运行一给定应用程序特定功能部件的访问级别。USER_LEVEL 具有的值最好为 ADMIN、REGULAR 和 RESTRICTED，其中 ADMIN 限制性最少，准许访问所有可用的应用程序功能部件，例如，检索报告，电子邮件，打印。RESTRICTED 限制用户至基础应用程序功能部件，例如，产生报告。REGULAR 级用户允许访问的功能部件比 ADMIN 级用户少，但比 RESTRICTED 级用户多。每个应用程序都可以基于它所提供的不同功能部件来解释 USER_LEVEL 属性。这就可以理解一些精确集中的属性，例如，与报告生成或打印有关，可以代替 USER_LEVEL 属性了。

两个属性 DATA_SCOPE 和 USER_LEVEL，以及它们各自的值，当然都只是例子。本领域技术人员可以理解定义限制访问应用程序功能部件的属性是不受限制的。

分组

属性可以分配给单独的用户，或者如最佳实施例中，执行分组的方案，其范例在图 2 中显示。属性如 DATA_SCOPE 和 USER_LEVEL 以方块表示，“属性组”以三角形表示，而“受托人组”以圆形表示。属性组只包含属性及它们的值，而受托人组包括属性组和/或其它的受托人组，并不是单个的属性。在首选的分组方案实例中，每个属性组被限定为对应一个应用程序的属性，从而提供可为不同的应用程序分配不同的属性和值的能力。在一个可供选择的实例中，属性组可以独立于特殊的应用程序而生成，这样一个组就可以包含所有应用程序的属性。尽管定义对单独应用程序功能部件的访问缺少了一些适应性，但这样的系统是比较容易实现的。

在这首选的实施例中，表 APPS，它至少包含一列为 APP_CODE，定义有相关属性的合法应用程序的列表。回到前面参考图 1，合法的 APP_CODE 值是 ABC、DEF 和 GHI。在 APPS 表中的其它列将包含每个应用程序所必需的任何信息。例如，在一菜单系统中，它提供图标，用户可以从中选择应用程序运行，APP_NAME 列将包含特殊字符串作为图标的可视标识它与 APP_CODE 列中的应用程序相关联。

每个属性组定义为由 APP_CODE 确定的特殊应用程序定义一组零或多个属性。零属性的属性组用来说明应该为应用程序分配属性的默认值。通常默认值是最具限制性的。另一种情况，零属性的属性组可以用来说明除了运行程序的能力，没有 APP_CODE 确定的访问被定义的应用程序功能部件。

图 2 的分组范例描绘了比前面讨论过与图 1 和表 1 有关的例子更复杂的用户结构。图 2 中属性组 106、115、130、145 和 148 属性分配如下：

表 2

属性组	应用程序	属性名称	属性值
106	GHI	DATA_SCOPE	ALL
		USER_LEVEL	ADMIN
115	DEF	DATA_SCOPE	ALL
		USER_LEVEL	REGULAR
130	ABC	DATA_SCOPE	A
		DATA_SCOPE	B
		USER_LEVEL	REGULAR
145	ABC	DATA_SCOPE	A
		DATA_SCOPE	B
		USER_LEVEL	RESTRICTED
148	DEF	DATA_SCOPE	B
		DATA_SCOPE	C
		USER_LEVEL	RESTRICTED

在这最佳实施例中，每个属性组对应的应用程序在该组建立的时候被指定，并在讨论的范例中由上面表 2 中第二列显示出来。

回到图 2，分配了属性组 106 的用户将有属性 DATA_SCOPE 107 和 USER_LEVEL 108，及其各自的值 ALL 和 ADMIN，如表 2 所示。根据属性，在程序被运行时，用户将被允许对应用程序 GHI 进行 ADMIN 式访问，并且被准许访问与客户 A、B 和 C 有关的数据。分配了属性组 130 的用户将有三个属性 DATA_SCOPE 133、DATA_SCOPE 136 和 USER_LEVEL139，以及各自的值 A、B 和 REGULAR。根据属性，这个用户可以访问与客户 A 或 B 相关的数据，并可以访问应用程序 ABC 的功能部件，它是为 REGULAR 身份的用户预先定义的。分配了属性组 145 的用户可以对应用程序 ABC 进行 RESTRICTED 式访问，并被准许访问与客户 A 和 B 有关的数据。分配了属性组 148 的客户将被允许对应用程序 DEF 进行 RESTRICTED 式访问，并被准许访问与客户 B 和 C 有关的数据。

一个或多个属性组可以分配给受托人组。图 2 中，例如受托人组 142 可以由属性组 145 和 148 组成，并且可以包括下面表 3 中最后两

列所列出的所有的属性名称-值对：

表 3

受托人组	属性组	应用程序	属性名称	属性值
142	145	ABC	DATA_SCOPE	A
			DATA_SCOPE	B
			USER_LEVEL	RESTRICTED
	148	DEF	DATA_SCOPE	B
			DATA_SCOPE	C
			USER_LEVEL	RESTRICTED

受托人组也可以分配给其它的受托人组。这可以从图 2 中受托人组 103 看出，它包括了受托人组 142 的所有属性以及属性组 130 和 135 的所有属性。在图上端，受托人组 100 由受托人组 103 和属性组 106 组成。因此，受托人组 100 包含了五个属性组 130、145、148、115 和 106 的所有属性。

属性分组系统特别适用于给责任级别不同的雇员分配属性。例如，属性组 115、130、145 和 148 可以分配给低或中级雇员，而受托人组 100、103 和 142 以及属性组 106 可以分配给管理人员，其职责是监督下级雇员的工作，至于属性组 106，就是运行他们自己的应用程序。

在优化的分组系统中，对属性组、受托人组和用户进行的属性分配保存在表 ATTRIBUTES 中。ATTRIBUTES 表有三列：ASSIGNEE、ATTRIBUTE_NAME 和 ATTRIBUTE_VALUE。ASSIGNEE 可以是属性组名称，受托人组名称或是用户。ATTRIBUTE_NAME 是属性名称（例如，DATA_SCOPE）。ATTRIBUTE_VALUE 是命名属性的特定值（例如，ALL）。

ATTRIBUTES 表由七个基本命令来维护。范例命令是以用于 Oracle 数据库环境中的 Oracle 程序写出的。本领域的技术人员可以导出适于其它环境的类似命令。在以下的描述中，单引号中是参数，双引号中是字符串。

命令 1

```
attr_utils.create_group('group_name','group_type','app
_code')
```

'group_name': 组的名称

'group_type': "ATTRIBUTE"或"ASSIGNEE"

'app_code': 如果'group_type'是"ATTRIBUTE",这一字段是需要的;否则,它将被忽略

这个例程将生成一指定类型的组。如果'group_name'已经作为一个组或 Oracle 用户存在,例程将由于错误而退出。

根据例程,'group_name'和'group_type'值转换成大写。然后,有下面一列值的记录被插入 ATTRIBUTES 表:

设置 ASSIGNEE = 'group_name'

设置 ATTRIBUTE_NAME = "ASSIGNEE_TYPE"

设置 ATTRIBUTE_VALUE = "ATTRIBUTE_GROUP" 或 "ASSIGNEE_GROUP"基于'group_type'的变量值。

另外,如果'group_type'是"ATTRIBUTE",下面有另一列值的记录将被插入 ATTRIBUTES 表:

设置 ASSIGNEE = 'group_name'

设置 ATTRIBUTE_NAME = "APP_CODE"

设置 ATTRIBUTE_VALUE = 'app_code'。

命令 2

attr_utils.assign_group('assignee','group_name')

'assignee': 有分配了'group_name'的用户或受托人组。

'group_name': 分配给受托人的组。

这个例程将把一个组分配给另外一个组或用户。如果'assignee'作为受托人组或用户不存在,将返回错误信息。如果'group_name'不存在,同样返回错误信息。

这个例程首先将'assignee'和'group_name'的值转换化大写。其次,有下面一列值的记录被插入 ATTRIBUTES 表:

设置 ASSIGNEE = 'assignee'

设置 ATTRIBUTE_NAME = "ASSIGNED_GROUP"

设置 ATTRIBUTE_VALUE = 'group_name'。

命令 3

attr_utils.assign_attribute('assignee','attribute_name','attribute_value)

'assignee': 属性组名称。这必须是一个属性组。

'attribute_name': 属性名称

'attribute_value': 指定属性的值

这个例程将分配值的属性给'assignee'。当受托人组作为一个属性组不存在，或是 attribute_name 是保留字段，将返回错误信息。

这个例程首选将'assignee'和'attribute_name'值转化成大写，然后再将有下面一列值的记录插入 ATTRIBUTES 表:

设置 ASSIGNEE = 'assignee'

设置 ATTRIBUTE_NAME = 'attribute_name'

设置 ATTRIBUTE_VALUE = 'attribute_value'。

命令 4

attr_utils.drop_group('group_name')

'group_name': 将与其相关信息一起被删除的组或用户的名称。

这个例程将删除组或用户及其相关信息。如果'group_name'不存在，将返回错误信息。

这个例程首先将'group_name'转化成大写，然后删除 ATTRIBUTES 表中 ASSIGNEE 列的值与'group_name'相符的所有记录。另外在 ATTRIBUTES 表中满足下面两个条件的所有记录也将被删除:

a. ATTRIBUTE_NAME 列的值是"ASSIGNED_GROUP"。

b. ATTRIBUTE_NAME 列的值与'group_name'相符。

命令 5

attr_utils.rescind_group('assignee','group_name')

'assignee': 用户或受托人组，其含有被删除的 group_name

'group_name': 从受托人组中删除的组

这个例程将从'assignee'中删除指定的'group_name'。如果'group_name'或'assignee'不存在，将返回错误信息。

这个例程首选将'assignee'和'group_name'转化成大写，然后将 ATTIBUTES 表中与下面三个条件符合的所有记录删除:

a. ATTIBUTE_NAME 列中的值是"ASSINGED_GROUP"。

b. ATTIBUTE_VALUE 列中的值与'group_name'相符。

c. ASSIGNEE 列中的值与'assignee'相符。

命令 6

attr_utils.rescind_attribute('assignee','attribute_name')

'assignee': 属性组名称。这必须是一个属性组。

'attribute_name': 属性名称。

这个例程将从'assignee'中删除指定的'attribute_name'。如果 attribute_name 和 assignee 不存在，或是 attribute_name 为保留字段，将返回错误信息。

这个程序首先将'assignee'和'attribute_name'转化成大写，然后将 ATTRIBUTES 表中满足以下两个条件的所有记录删除:

a. ASSIGNEE 列中的值与'assignee'相符。

b. ATTRIBUTE_NAME 列中的值与'attribute_name'相符。

命令 7

attr_utils.update_attribute('assignee','attribute_name','attribute_value')

'assignee': 属性组名称。这必须是一个属性组。

'attribute_name': 属性名称。

'attribute_value': 指定属性的新值。

这个例程将为确定的'assignee'和'attribute_name'更新指定的'attribute_value'。如果 attribute_name 或 assignee 不存在，或是 attribute_name 为保留字段，将返回错误信息。

这个例程首先将'assignee'和'attribute_name'转化成大写，然后更新 ATTRIBUTES 表，对于满足以下两个条件的所有记录，将把 ATTRIBUTE_VALUE 设置成'attribute_value':

a. ASSIGNEE 列中的值与'assignee'相符。

b. ATTRIBUTE_NAME 列中的值与'attribute_name'相符。

利用前面的命令，就维护了一个 ATTRIBUTES 表。如命令 1 和 2 中所示，在首选实施例中，一些保留的 ATTRIBUTE_NAME 在 ATTRIBUTES 表中使用，用来确定系统中所用到的特殊信息。"APP_CODE"的 ATTRIBUTE_NAME 自动分配给属性组，用以确定与组相关的应用程序。"ASSIGNED_GROUP"的 ATTRIBUTE_NAME 是用来给属性组分配属性，给受托人组分配属性组和受托人组，还给用户分配属性组和受托

人组。"ASSIGNEE_TYPE"的 ATTRIBUTE_NAME 用来确定一个组是属性组还是受托人组。这些例程检查 ATTRIBUTE_NAME 参数以确定它们不是保留字段，如果使用了保留的 ATTRUBUTE_NAME 将返回错误信息。

举例说明，在下面的表 4 中将显示图 2 中受托人组 142 的 ATTRIBUTES 表的一部分：

表 4

受托人	ATTRIBUTE_NAME	ATTRIBUTE_VALUE
ASSIGNEE_GOUP_142	ASSIGNEE_TYPE	ASSIGNEE_GROUP
ATTRIBUTE_GROUP_145	ASSIGNEE_TYPE	ATTRIBUTE_GROUP
ATTRIBUTE_GROUP_145	APP_CODE	ABC
ATTRIBUTE_GROUP_148	ASSIGNEE_TYPE	ATTRIBUTE_GROUP
ATTRIBUTE_GROUP_148	APP_CODE	ABC
ATTRIBUTE_GROUP_145	DATA_SCOPE	A
ATTRIBUTE_GROUP_145	DATA_SCOPE	B
ATTRIBUTE_GROUP_145	USER_LEVEL	RESTRICTED
ATTRIBUTE_GROUP_148	DATA_SCOPE	B
ATTRIBUTE_GROUP_148	DATA_SCOPE	C
ATTRIBUTE_GROUP_148	USER_LEVEL	RESTRICTED
ASSIGNEE_GOUP_142	ASSIGNED_GROUP	ATTRIBUTE_GROUP_142
ASSIGNEE_GOUP_142	ASSIGNED_GROUP	ATTRIBUTE_GROUP_145

在本发明供选择的实施例中，前面所讨论过的，属性组不是限制于特定的应用程序，creat_group 例程不一定要求有'app_code'输入，而 APP_CODE 属性将不保存在 ATTRIBUTES 表中。但是，利用附加表将 APP_CODE 直接分配给用户，对应用程序大规模的访问仍可以控制。如果用户没有分配到特定的 APP_CODE，用户将完全不能获得相应的应用程序。

父子结构

为单一的受托人组分配多重属性组和/或受托人组导致可以对同一组或用户分配交迭、重复甚至冲突的值。例如，受托人组 100，图 2 上端所示，包括图中的每个属性，因此，如前面表 2 所示，对于同一应用程序 DEF，DATA_SCOPE 属性包括不同的值 B、C 和 ALL，而

USER_LEVEL 有不同的值 REGULAR 和 RESTRICTED。由于这个原因，在首选的实施例中，建立属性分层结构，其中每个属性值都分配了“父值”。例如，DATA_SCOPE 属性的值 B 分配了父值 ALL。在实际中，这被称作用户分配的属性，同一属性和应用程序的父值和子值都存在，父值将被保存而子值被删除。此外，重复的值将被删除。

父子的分配被保存在 ATTRIBUTE_LEVELS 表中，其中包括三列：ATTRIBUTE_NAME、CHILD_VALUE 和 PARENT_VALUE。ATTRIBUTE_NAME 是属性的名称（例如，DATA_SCOPE）。CHILD_VALUE 是属性的实际值（例如，C）。对于 PARENT_VALUE 来说，实际值是它的子集（例如，ALL）。有种情况，即实际值是结构中级别最高的，例如 ALL，所分配的父值就是 NULL（空）。

ATTRIBUTE_LEVELS 表由四个基本命令维护。范例命令是以用于 Oracle 数据库环境中的 Oracle 程序写出的。本领域的技术人员可以导出适于其它数据库环境的类似命令。注意在以下的描述中，单引号中是参数，双引号中是字符串。

命令 1

attr_utils.add_attr_level('attribute_name','child_value','parent_value')

'attribute_name'：属性名称

'child_value'：对应指定'attribute_name'的子值

parent_value'：对应指定'attribute_name'，指定'child_value'的父值

这个例程将给一指定的指针加上新的属性级别。如果'child_value'在指定的'attribute_name'的权限内具有最高级别，则'parent_value'为"NULL"。如果父值为空或不存在，将返回错误信息。

这个例程首先将'attribute_name'转化成大写，然后将有下面一列值的记录插入 ATTRIBUTE_LEVELS 表：

设置 ATTRIBUTE_NAME ='attribute_name'

设置 CHILD_VALUE ='child_value'

设置 PARENT_VALUE ='parent_value'

命令 2

attr_utils.update_attr_level('attribute_name','child_value', 'parent_value')

'attribute_name': 属性名称

'child_value': 对应指定'attribute_name'的子值

parent_value': 对应指定'attribute_name', 指定'child_value'的父值

这个例程将为指定的参数更新父值。如果'child_value'在指定的'attribute_name'的权限内具有最高级别，则'parent_value'为"NULL"。如果父值为空或不存在，或者 attribute_name 与 child_value 组合不存在，将返回错误信息。

这个例程首先将'attribute_name'转化成大写，然后更新 ATTRIBUTE_LEVELS 表，对于满足下面两个条件的所有记录，将 PARENT_VALUE 列的值设置为'parent_value':

a. ATTRIBUTE_NAME 列的值与'attribute_name'相符。

b. CHILD_VALUE 列的值与'child_name'相符。

命令 3

attr_utils.delete_attr_levels('attribute_name','child_value')

'attribute_name': 属性名称

'child_value': 对应指定'attribute_name'的子值

这个例程将删除属性级别以及指定指针所有的子属性级别。如果组合不存在，将返回错误信息。

这个例程首先将'attribute_name'转化成大写，然后将 ATTRIBUTE_LEVELS 表中指定组合对'attribute_name'和'child_value'所派生出的所有记录都删除。举例说明，以下 SQL 语句可以用来完成头两步:

```
DELETE ATTRIBUTE LEVELS
WHERE (ATTRIBUTE_NAME,CHILD_VALUE) IN
(SELECT ATTRIBUTE_NAME,CHILD_VALUE
FROM ATTRIBUTE_LEVELS
START WITH PARENT_VALUE=P_CHILD_VALUE
AND ATTRIBUTE_NAME=UPPER(P_ATTRIBUTE_NAME)
```

CONNECT BY PARENT_VALUE=PRIOR CHILE_VALUE

AND ATTRIBUTE_NAME=PRIOR ATTRIBUTE_NAME).

这个例程将删除 ATTRIBUTE_LEVELS 表中满足以下两个条件的所有记录:

a. ATTRIBUTE_NAME 列的值与'attribute_name'相符。

b. CHILD_VALUE 列的值与'child_name'相符。

命令 4

attr_utils.delete_all_levels('attribute_name')

'attibute_name': 属性名称

这个例程将删除指定 attribute_name 的所有属性级别。如果 attribute_name 不存在，将返回错误信息。

这个例程首先将'attribute_name'转化成大写，然后将 ATTRIBUTE_LEVELS 表中 ATTRIBUTE_NAME 列值与'attribute_name'相符的所有记录删除。

利用前面的命令，ATTRIBUTE_LEVELS 表被保存。举例说明，下面的表 5 显示了一个 ATTRIBUTE_LEVELS 表，是对应前面讨论过的范例属性:

表 5

ATTRIBUTE_NAME	CHILD_VALUE	RARENT_VALUE
DATA_SCOPE	A	ALL
DATA_SCOPE	B	ALL
DATA_SCOPE	C	ALL
DATA_SCOPE	ALL	NULL
USER_LEVEL	RESTRICTED	REGULAR
USER_LEVEL	REGULAR	ADMIN
USER_LEVEL	ADMIN	NULL

根据表 5，如果对于同一应用程序，拥有值 ALL 的 DATA_SCOPE 赋值给同一用户，对于拥有值 A、B 或 C 的 DATA_SCOPE 属性的赋值将被删除。同样，较低级 USER_LEVEL 值的赋值将被删除以利于最高级别值的赋值。

更进一步，如果属性值在 ATTRIBUTES 表中赋值了，而没有在

ATTRIBUTE_LEVELS 表中定义，则被认为在 ATTRIBUTE_LEVELS 表中它被定义为带有 NULL 父值，并且没有具有值的其它值作为它的父值。

考虑到较低级完整的一组赋值变量可以由较高级变量取代，进一步的简化就可以实现了。例如，DATA_SCOPE 值 A、B 和 C 被指定了，系统可以返回值 ALL。这种简化只能在所有的较低级值与较高级值表示同一事物时才可使用，因为较高级值可能比较低级值代表的要多一些。

用户属性系统的操作

结合以下对本发明的用户属性系统示范使用的讨论，将对图 3 作出参考说明。根据发明的首选实施例，对数据库环境中运行程序的访问由初始的图形用户界面(IGUI)控制。IGUI 的例子包括互联网站点主页和局域网的初始页面。

但是，在获准访问应用程序之前，用户通常要登录到计算机系统中，方框 201 所示，如果登录完全正确的话，计算机系统将确认用户。方框 204，IGUI 检索用户可用的应用程序。在首选实施例中，IGUI 通过读取 ATTRIBUTES 表做到这点的，箭头 205 包含了给用户的分组分配。如前面所讨论的，在首选实施例中，组分配包括属性组分配，而它又包含了可用程序的指定。在 Oracle 数据库系统中准备了“视图（view）”，它优于搜索整个 ATTRIBUTES 表，而表又可能相当庞大，它保存有频繁使用的搜索结果（例如，特殊用户的属性），而且很有可能被反复使用。在可选择的实例中，属性不是限制于特定的应用程序，特殊用户可用的应用程序可以在分离的表中存储和检索。

在箭头 205，IGUI 也将访问合法程序的 APPS 表，其包含的 APP_CODE 列告知 IGUI 在方框 207 中显示给用户哪些字符串，此方框中 IGUI 显示给用户可用的应用程序。方框 210 中，用户选择其中一个可用的应用程序。方框 213 中，IGUI 读取 ATTRIBUTES 表，在箭头 214，检索对应所选程序的用户属性。在 Oracle 数据库系统中，视图可以再一次用于检索属性。另外，IGUI 可以访问 ATTRIBUTE_LEVELS 表来删除属性个数。

在箭头 215，IGUI 将相关的属性传给应用程序，而在方框 216 中，应用程序，按程序执行属性运行。如果用户试图超越指定的限制，可能会显示错误或警告信息。

在可选择的实例中，本领域的技术人员可以理解可以直接检索属性，不需要 IGUI 的辅助，并执行属性。

视图

如前面所讨论的，Oracle 视图在本发明中用于检索和组织来自表的记录。以下是十二个视图的列表，它们在本发明的执行中有用。这里提供了每个视图的解释和一个 SQL 范例。

视图 1

V_ATTRIBUTE_APP_CODES

这个视图将返回，用户属性系统中安装的所有不同的应用程序代码的列表。这个视图对记录执行一条含有 DISTINCT 子句的 SELECT，记录中的 ATTRIBUTE_NAME 是'APP_CODE'保留的 ATTRIBUTE_NAME。

SQL 范例:

```
CREATE OR REPLACE VIEW V_ATTRIBUTE_APP_CODES AS
SELECT DISTINCT ASSIGNEE,
ATTRIBUTE_VALUE_APP_CODE
FROM ATTRIBUTES
WHERE ATTRIBUTE_NAME='APP_CODE'
```

视图 2

V_ATTRIBUTE_GROUPS_ATTR

这个视图将返回所有不同的属性组列表。这个视图对记录执行一条含有 DISTINCT 子句的 SELECT，记录中的 ATTRIBUTE_NAME 是'ASSIGNEE_TYPE'保留的 ATTRIBUTE_NAME，而 ATTRIBUTE_VALUE 是'ATTRIBUTE_GROUP'。

SQL 范例:

```
CREATE OR REPLACE VIEW V_ATTRIBUTE_GROUPS_ATTR
AS SELECT DISTICT ASSIGNEE GROUP_NAME
FROM ATTRIBUTES
WHERE ATTRIBUTE_NAME='ASSIGNEE_TYPE'
AND ATTRIBUTE_VALUE='ATTRIBUTE_GROUP';
```

视图 3

V_ATTRIBUTE_GROUPS_ASSIGN

这个视图将返回所有不同的受托人组列表。这个视图对记录执行

一条含有 DISTINCT 子句的 SELECT，记录中的 ATTRIBUTE_NAME 是'ASSIGNEE_TYPE'保留的 ATTRIBUTE_NAME，而 ATTRIBUTE_VALUE 是'ATTRIBUTE_GROUP'。

SQL 范例：

```
CREATE OR REPLACE VIEW
V_ATTRIBUTE_GROUPS_ASSIGN AS
SELECT DISTINCT ASSIGNEE GROUP_NAME
FROM ATTRIBUTES
WHERE ATTRIBUTE_NAME='ASSIGNEE_TYPE'
AND ATTRIBUTE_VALUE='ASSIGNEE_GROUP';
```

视图 4

V_ATTRIBUTE_USERS

这个视图将返回所有不同的属性用户列表。这个视图对记录执行一条含有 DISTINCT 子句的 SELECT，记录中的 ASSIGNEE 等于在 Oracle 数据程序库表 ALL_USERS 中找到的 USER_NAME。

SQL 范例：

```
CREATE OR REPLACE VIEW V_ATTRIBUTE_USERS AS
SELECT DISTINCT ASSIGNEE USERID
FROM ATTRIBUTES,
ALL_USERS
WHERE ASSIGNEE=USERNAME;
```

视图 5

V_ATTRIBUTE_GROUPS_ALL

这个视图将返回系统中所有不同的组列表。这包括属性组和受托人组。这个视图对记录执行一条含有 DISTINCT 子句的 SELECT，记录中的 ASSIGNEE 不是属性用户。

SQL 范例：

```
CREATE OR REPLACE VIEW V_ATTRIBUTE_GROUPS_ALL AS
SELECT DISTINCT ASSIGNEE GROUP_NAME
FROM ATTRIBUTES,
V_ATTRIBUTE_USERS
WHERE ASSIGNEE=USERID(+)
```

```
AND USERID IS NULL;
```

视图 6

V_USER_GROUPS

这个视图将返回分配给当前连接到 Oracle 用户的所有组的列表。这个结果包括直接分配给用户的组和间接分配给用户的组。进一步说，就是组分配给 ASSIGNEE GROUPS，而 ASSIGNEE GOUPS 又分配给了用户。这个视图执行使用 CONNECT BY 子句的系统树型查询。

SQL 范例:

```
CREATE OR REPLACE VIEW V_USER_GROUPS AS
    SELECT ATTRIBUTE_NAME,
    ATTRIBUTE_VALUE
    FROM ATTRIBUTES
    WHERE ATTRIBUTE_NAME!='ASSIGNEE_TYPE'
    START WITH ASSIGNEE=USER
    CONNECT BY ASSIGNEE=PRIOR ATTRIBUTE_VALUE
    AND ATTRIBUTE_NAME='ASSIGNED_GROUP';
```

视图 7

V_USER_ATTR_APPS

这个视图返回一列表，列表中是分配给当前连接到 Oracle 用户的所有属性，及相应的 APP_CODE。这个视图将把分配给用户的组（V_USER_GROUP）列表，ATTRIBUTES 表，和带有相应 APP_CODE 变量（V_ATTRIBUTE_APP_CODES）的 ATTRIBUTE_GROUPS 列表结合起来。

SQL 范例:

```
CREATE OR REPLACE VIEW V_USER_ATTR_APPS AS
    SELECT ATTR.ATTRIBUTE_NAME,
    ATTR.ATTRIBUTE_VALUE,
    APPS.APP_CODE
    FROM V_USER_GROUPS GROUPS,
    ATTRIBUTES ATTR,
    V_ATTRIBUTE_APP_CODES APPS
    WHERE GROUPS.ATTRIBUTE_VALUE=ATTR.ASSIGNEE
    AND ATTR.ASSIGNEE=APPS.ASSIGNEE
```

```
AND ATTR.ATTRIBUTE_NAME NOT IN
('ASSIGNED_GROUP','APP_CODE','ASSIGNEE_TYPE');
```

视图 8

V_USER_ATTR_HIGHEST_VALUES

这个视图将返回对于相应的 ATTRIBUTE_NAME 来说级别最高的 ATTRIBUTE_VALUE 列表。这个视图可以包含重复的条目，这样在后面的讨论中，V_USER_ATTRIBUTES 视图可以检索这些不同值的列表。这个视图将会为每个分配给当前用户的属性，把 APP_CODE、ATTRIBUTE_NAME 和 ATTRIBUTE_VALUE 传递到 ATTR_UTILS.HIGHEST_VALUE 函数中。函数与程序相同，只是它可以作为查询的一部分执行，并返回一个值。

这里，函数返回当前分配给用户的最高级父值。在此视图 SQL 范例之后是函数描述。

SQL 范例:

```
CREATE OR REPLACE VIEW
V_USER_ATTR_HIGHEST_VALUES AS
SELECT APP_CODE,
ATTRIBUTE_NAME,
SUBSTR(ATTR_UTILS.HIGHEST_VALUE(APP_CODE,
ARRTIBUTE_NAME,ATTRIBUTE_VALUE),1,30)
ATTRIBUTE_VALUE
FROM V_USER_ATTR_APPS;
```

ATTR_UTILS.HIGHEST_VALUE 函数要求三个输入: APP_CODE, ATTRIBUTE_NAME 和 ATTRIBUTE_VALUE。函数要先载入内部属性值表，此表由为给定用户分配的所有属性值组成，APP_CODE 和 ATTRIBUTE_NAME 。函数可以利用一现成的视图（即 V_USER_ATTR_VALUE_LEVELS）完成这一工作。然后函数将使用当前的属性值，并找到这一值的所有父记录。函数将使用如下的 SQL 查询:

```
SELECT PARENT_VALUE
FROM ATTRIBUTE_LEVELS
WHERE ATTRIBUTE_NAME=P_ATTRIBUTE_NAME
```

START WITH CHILD_VALUE=P_ATTRIBUTE_VALUE
CONNECT BY CHILD_VALUE=PRIOR PARENT_VALUE
ORDER BY LEVEL;

然后这个函数将每个父值与属性值表中的记录进行比较，以确定其它分配了属性的值中是否有父值，即当前属性值的父级，祖父级，及其它。如此后，最高级的属性值被返回。否则当前属性值被返回。

视图 9

V_ATTR_VALUE_LEVELS

这个视图将返回一列表，包括 ATTIBUTE_NAME、相关的 ATTRIBUTE_VALUE 和变量相应的级别。例如，最高级 ATTIBUTE_VALUE 将拥有值为 1 的 ATTRIBUTE_LEVEL，而值的子级将拥有值 2。这个视图执行使用 CONNECT BY 子句的系统树型查询。

SQL 范例:

CREATE OR REPLACE VIEW V_ATTR_VALUE_LEVELS AS
SELECT ATTRIBUTE_NAME,
CHILD_VALUE ATTRIBUTE_VALUE,
LEVEL VALUE_LEVEL
FROM ATTRIBUTE_LEVELS
START WITH PARENT_VALUE IS NULL
CONNECT BY PARENT_VALUE=PRIOR CHILD_VALUE
AND ATTRIBUTE_NAME=PRIOR ATTRIBUTE_NAME;

视图 10

V_USER_ATTR_VALUE_LEVELS

这个视图将返回所有分配给用户的属性及其相应级别的列表。这个视图将把分配给当前连接 Oracle 的用户属性的列表（V_USER_ATTR_APPS）和属性与其相应级别的列表（V_ATTR_VALUE_LEVELS）结合起来。

SQL 范例:

CREATE OR REPLACE VIEW
V_USER_ATTR_VALUE_LEVELS
AS SELECT APP_CODE,
ATTR.ATTRIBUTE_NAME,

```
ATTR.ATTRIBUTE_VALUE,
NVL(VALUE_LEVEL,1) VALUE_LEVEL
FROM V_USER_ATTR_APPS ATTR,
V_ATTR_VALUE_LEVELS LVL
WHERE
ATTR.ATTRIBUTE_NAME=LVL.ATTRIBUTE_NAME(+)
AND
ATTR.ATTRIBUTE_VALUE=LVL.ATTRIBUTE_VALUE(+);
```

视图 11

V_USER_APP_CODES

这个视图将返回所有各别的分配给当前用户的 APP_CODE 列表。这个视图执行使用 CONNECT BY 子句的系统树型查询。

SQL 范例:

```
CREATE OR REPLACE VIEW V_USER_APP_CODES AS
SELECT DISTINCT ATTR.ATTRIBUTE_VALUE APP_CODE
FROM(SELECT ATTRIBUTE_NAME,
ATTRIBUTE_VALUE
FROM ATTRIBUTES
START WITH ASSIGNEE=USER
CONNECT BY ASSIGNEE=PRIOR ATTRIBUTE_VALUE
AND ATTRIBUTE_NAME='ASSIGNED_GROUP') GROUPS,
ATTRIBUTES ATTR
WHERE GROUPS.ATTRIBUTE_VALUE=ATTR.ASSIGNEE
AND ATTR.ATTRIBUTE_NAME='APP_CODE';
```

视图 12

V_USER_ATTRIBUTES

这个视图返回一列表，表中为所有分配给用户的属性，和只是相应 ATTRIBUTE_NAME 的最高级 ATTRIBUTE_VALUE。这个视图在 V_USER_ATTR_HIGHEST_VALUES (V_USER_ATTR_VALUE_LEVELS) 上执行 SELECT DISTINCT。

SQL 范例:

```
CREATE OR REPLACE VIEW V_USER_ATTRIBUTES AS
```

```
SELECT DISTINCT
APP_CODE,
ATTRIBUTE_NAME,
ATTRIBUTE_VALUE
FROM V_USER_ATTR_HIGHEST_VALUES;
```

在 Oracle 环境中，为了确定将 ATTRIBUTES 表和 ATTRIBUTE_LEVELS 表中的数据正确存储，最好是只用 Oracle 程序执行所有对用户属性的存储。这是可以保证的，通过在用户属性系统内限制访问不同目标，如表、视图、程序和函数。只读权限应该给 ATTRIBUTES 表和 ATTRIBUTE_LEVELS 表，以及所有的视图。用于存储这些表的 Oracle 程序，其执行权限应该只给用户属性管理者。这样可以确定未被授权的用户不能操纵属性或属性级别。

本发明已经，参考了一些有说明性的范例和实例描述，但这些不能理解为是限制本发明的范围或是思想。在不背离所附权利要求中表述的发明范围的前提下，可以在实际应用中，由本领域的技术人员进行许多修改。

说 明 书 附 图

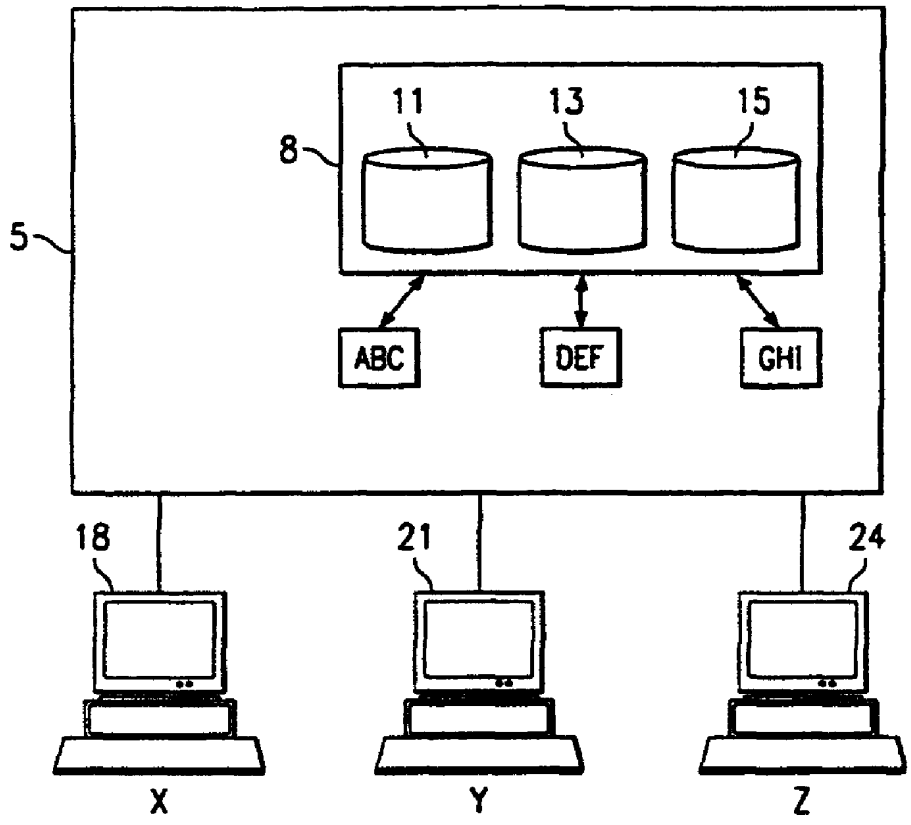

11
13
15
8
5
ABC
DEF
GHI
18
21
24
X
Y
Z

图 1

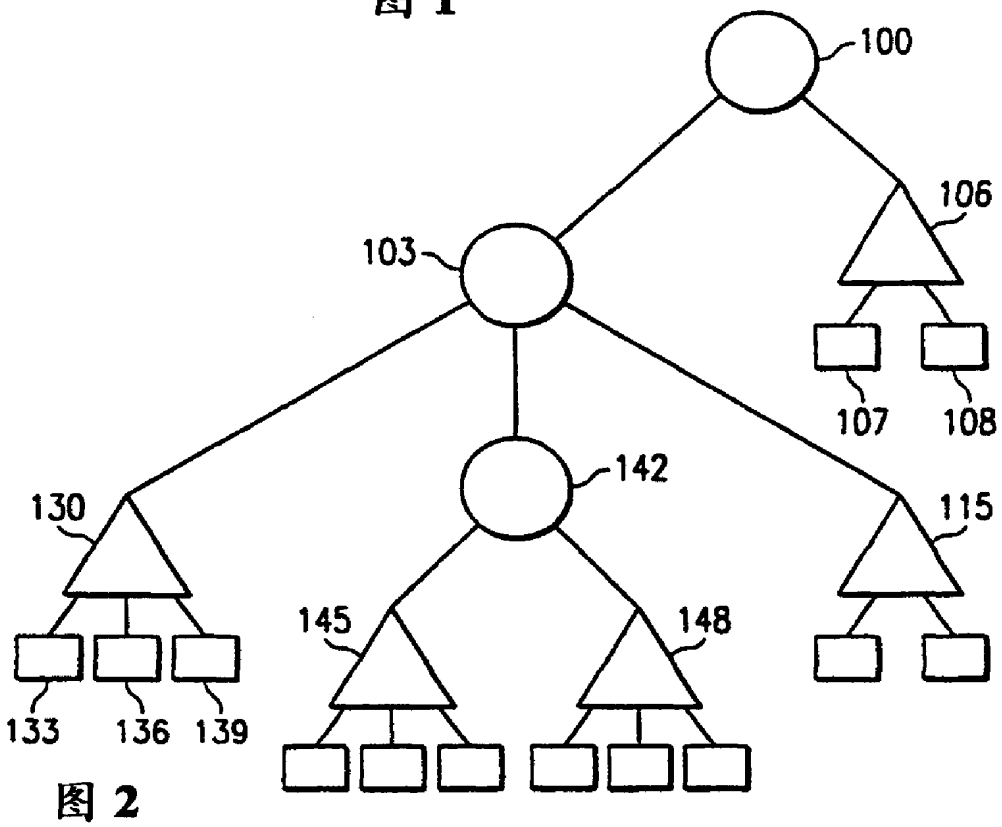

100
106
103
107
108
142
130
115
145
148
133
136
139

图 2

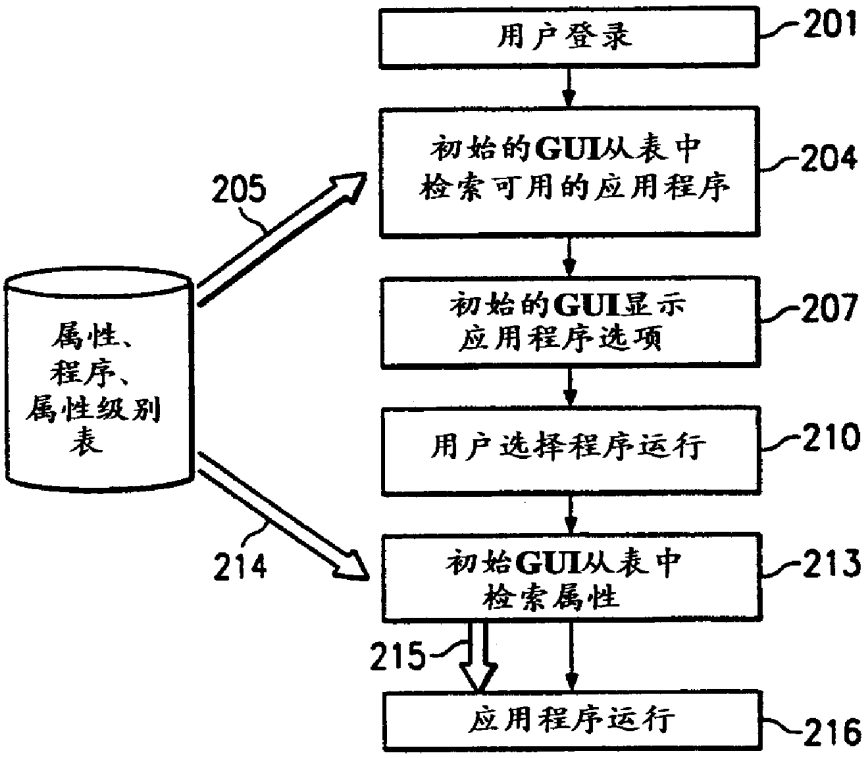

用户登录
201
初始的GUI从表中
检索可用的应用程序
204
205
初始的GUI显示
应用程序选项
207
属性、
程序、
属性级别
表
用户选择程序运行
210
214
初始GUI从表中
检索属性
213
215
应用程序运行
216

图 3